NPS-MA-92-004

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California


SOFTWARE FOR THE PARALLEL
CONSERVATIVE SCHEME FOR
THE SHALLOW WATER EQUATIONS

Levi Lustman
Beny Neta

February 1992

Approved for public release; distribution unlimited
Prepared for: Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

Rear Admiral R. W. West, Jr.
Superintendent

Harrison Shull
Provost

This report was prepared for and funded by the Naval Postgraduate School.



Prepared by:

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION / AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited |
| 2b DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S)<br>NPS-MA-92-004 | 5 MONITORING ORGANIZATION REPORT NUMBER(S)<br>NPS-MA-92-004 |
| 6a NAME OF PERFORMING ORGANIZATION<br>Naval Postgraduate School<br>6b OFFICE SYMBOL (If applicable)<br>MA | 7a NAME OF MONITORING ORGANIZATION<br>Naval Postgraduate School |
| 6c ADDRESS (City, State, and ZIP Code)<br>Monterey, CA 93943-5000 | 7b ADDRESS (City, State, and ZIP Code)<br>Monterey, CA 93943-5000 |
| 8a NAME OF FUNDING / SPONSORING ORGANIZATION<br>Naval Postgraduate School<br>8b OFFICE SYMBOL (If applicable)<br>MA | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>O&MN Direct Funding |
| 8c ADDRESS (City, State, and ZIP Code)<br>Monterey, CA 93943-5000 | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO |

11 TITLE (Include Security Classification)
SOFTWARE FOR THE PARALLEL CONSERVATIVE SCHEME FOR THE SHALLOW WATER EQUATIONS

12 PERSONAL AUTHOR(S)
Levi Lustman and Beny Neta

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| Technical REport | FROM 10/91 TO 12/91 | 14 February 1992 | 30 |

16 SUPPLEMENTARY NOTATION

| 17 COSATI CODES | | | 18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | parallel conservative scheme, shallow water equations, hypercube, finite difference scheme |
| | | | |
| | | | |

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report contains the host and node programs for the solution of the shallow water equations with topography on an INTEL iPSC/2 hypercube. Finite difference scheme conserving potential enstrophy and energy is employed in each subdomain.

| | |
|---|---|
| 20 DISTRIBUTION / AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a NAME OF RESPONSIBLE INDIVIDUAL<br>Beny Neta | 22b TELEPHONE (Include Area Code)<br>(408) 646-2235<br>22c OFFICE SYMBOL<br>MA/Nd |

DD Form 1473, JUN 86 *Previous editions are obsolete*
S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

# Software for the Parallel Conservative Scheme for the Shallow Water Equations

L. Lustman

and

B. Neta

Naval Postgraduate School
Department of Mathematics
Code MA/Nd
Monterey, CA 93943

**Abstract**

This report contains the host and node programs for the solution of the shallow water equations with topography on an INTEL iPSC/2 hypercube. Finite difference scheme conserving potential enstrophy and energy is employed in each subdomain.

## 1. Introduction

In this report we supply software for the numerical solution of the shallow water equations on an INTEL iPSC/2 hypercube. The method is based on domain decomposition with overlap. Finite difference scheme is used to solve in each subdomain. The scheme conserves potential enstrophy and energy (Arakawa C grid [1].) See also Neta and Lustman [2]. The efficiency of the algorithm is 81% when using 8 processors.

**2. Host program**

```
      PROGRAM HOST
c
      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
      parameter(lbuf=10*nprocs+30+lparm0,leninit=lbuf*4)
c                                        4 bytes per float
      parameter(inityp=914)
      parameter(nodes=-1,idhost=2,nodepid=3)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx-1)
      parameter(i9dim=4+(1+i9global)/nprocs)
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c
      parameter( myslv=1 , ibev=2 , iav=3 , i0gv=4)
      parameter( iminv=5 , imaxv=6 , i9v=7)

      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/uvh0com/u0,v0,top,width,h0,parm0(lparm0)

c
      parameter(lenuvh=4*(i9dim+1)*3*(j9+1))
      parameter(ku=1,kv=2,kh=3)

      logical done(0:node9),alldone
      dimension uvh(0:j9,3,0:i9dim)
      dimension buf(lbuf)
      dimension b00kp(10,0:node9)
      equivalence (b00kp,buf)



      call getcube('arakawa',' ',' ',1)
      call setpid(idhost)
      nproc=numnodes()
       print*,' got the maximal cube,',nproc,' nodes'
      call load('node',nodes,nodepid)

      print*,' domain parameters xmax,ymax,meshx,meshy='
     ,,maxx,maxy,meshx,meshy
      print*,' this generates a grid (0:',i9global
     ,,',0:',j9,')'
      do 13 i=0,node9
13      done(i)=.false.
      call bokeph(buf)
```

```
      l=10*nprocs+1
      call getdata
      buf(l)=dt
      l=l+1
      buf(l)=trepo
      l=l+1
      buf(l)=ttot
          nstep9=ttot/dt
c             the only parameter host must know is the maximal
c             number of steps, to decide whether the nodes
c             are done
c
      l=l+1
      buf(l)=coriof
      l=l+1
      buf(l)=corioa
      l=l+1
      buf(l)=coriob
      l=l+1
      buf(l)=g
      l=l+1
      buf(l)=u0
      l=l+1
      buf(l)=v0
      l=l+1
      buf(l)=top
      l=l+1
      buf(l)=width
      l=l+1
      buf(l)=h0
      l=l+1
      do 976 j=1,lparm0
      buf(l)=parm0(j)
      l=l+1
976      continue

      call csend(inityp,buf,leninit,nodes,nodepid)

c
c now the host will wait for results
c all come in uvh, but must be unscrambled using
c the message type to be printed
c

      n0=0
      open(UNIT=11,FORM='FORMATTED',FILE='ARA')
1132      continue
      call crecv(-1,uvh,lenuvh)
c                              any message at all
      ityp=infotype()
      n=ityp/100
```

```
c                       number of steps
      if(n.ge.n0) then
      write(UNIT=11,FMT=7500)n*dt,n*dt/3600,n*dt/24/3600
7500      format(' Report after ',f10.0,' sec =',f10.2,' hours ='
     ,,f10.2,' days')
      n0=n+1
c
c to print the title just once, not for each processor
c
      endif
      node=mod(ityp,100)
      imin=b00kp(iminv,node)
      imax=b00kp(imaxv,node)
      i0g=b00kp(i0gv,node)
c                       to convert i to global i
      do 76 i=imin,imax
      iglo=mod(i0g+i , i9global+1)
      do 76 j=0,j9
      write(UNIT=11,FMT=7600)n,j+1,iglo+1
     ,,uvh(j,ku,i),uvh(j,kv,i),uvh(j,kh,i)
76      continue
7600      format(i7,2i4,3g20.5)
      done(node)= (n.ge.nstep9)
452      continue
      alldone=done(0)
      do 23 i=1,node9
23      alldone=alldone.and.done(i)
      if(.not.all done) goto 1132
c                               more messages coming
c
c the test above replaces waitall, which cannot be used
c

      call relcube('arakawa')
      stop
      end
      subroutine bokeph (b00kp)
c
c each node sees its data as an array (0:j9,0:i9)
c
c the column (,i) in the node data is the same as (,i+i0global) in
c the full matrix, which is (0:j9,0:i9global)
c
      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx-1)
      parameter(i9dim=4+(1+i9global)/nprocs)
```

```
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c
      parameter( myslv=1 , ibev=2 , iav=3 , i0gv=4)
      parameter( iminv=5 , imaxv=6 , i9v=7)

      dimension b00kp(10,0:node9)
      integer gray,ginv
      integer i0w(0:node9)
      integer i9w(0:node9)


      linnod=(i9global+1)/nprocs
      do 1 i=0,node9
      i0w(i)=i*linnod
      i9w(i)=i0w(i)+linnod-1
1      continue
      iad=0
15      continue
      if(i9w(node9).ne.i9global) then
      i9w(iad)=i9w(iad)+1
      do 2 i=iad+1,node9
      i0w(i)=   i0w(i)+1
      i9w(i)=   i9w(i)+1
2      continue
      iad=iad+1
      goto 15
      endif
      print*,i0w
      print*,i9w
c
c at this stage, i0w(slice) to i9w(slice) are the columns that
c belong to slice, and will be advanced in time by
c the processor that treats slice
c
c but it may need four additional columns, to compute
c neighborhood averages
c
      do 333 iam=0,node9
      myslice=ginv(iam)
      i0wm=i0w(myslice)-2
      i0m=i0wm
      if (i0wm.lt.0) i0wm=i0wm+i9global+1
      i0w(myslice)=i0wm
      i9wm=i9w(myslice)+2
      i9m=i9wm
      if (i9wm.gt.i9global) i9wm=i9wm-i9global-1
      i9w(myslice)=i9wm
      i9=i9m-i0m
      ibefore=-1
      iafter=-1
```

```
      myp=mod(3*nprocs+myslice+1,nprocs)
      mym=mod(3*nprocs+myslice-1,nprocs)
      iafter=gray(myp)
      ibefore=gray(mym)
c=================================================
c
c      also define imin, imax
c these are the lines which this node should advance in time,
c the other lines are for information only
c
      imin=2
      imax=i9 -2
c=================================================
      b00kp(i0gv,iam)=i0w(myslice)
      b00kp(myslv,iam)=myslice
      b00kp(ibev,iam)=ibefore
      b00kp(iav,iam)=iafter
      b00kp(i9v,iam)=i9
      b00kp(imaxv,iam)=imax
      b00kp(iminv,iam)=imin
      print*,'    ..............    I am ',iam
      print*,' my slice is ', myslice
      print*,' procs. before,after me=',ibefore,iafter
      print*,' my data have dim (,0:' ,i9,')'
     ,,imin, ' to ',imax,' meaningful'
      iii=b00kp(i0gv,iam)
      print*,' my column 0 is global column ',iii
333      continue
      return
      end
      subroutine getdata

      parameter(lparm0=10)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)

      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/uvh0com/u0,v0,top,width,h0,parm0(lparm0)
      character*3 name

      minmsh=min(meshx,meshy)
      dt=150.*minmsh/125
      ttot=24*3600*5
c                     5 days
      trepo= -ttot
c
c             silly programming trick, so ttot,trepo
c             may be entered in any order
c
      coriof=1.e-4
      corioa=0
```

```
      coriob=0
      g=9.8e-3
c                       notice! length in km !
      u0=20e-3
      v0=0
      top=2
      width=1000
      h0=5
      do 1 i=1,lparm0)
1      parm0(i)=0
c
c all these are defaults. then more data may be read
c
1132      continue
      read100,name
100      format(a3)
      if(name.eq.'par') then
1131      read*,i,x
      if(i.gt.0.and.i.le.lparm0) then
                        parm0(i)=x
                        goto 1131
              else
                        goto 1132
              endif
      else
              if(name.eq.'end') goto 9999
              if(name.eq.'sto') goto 9999
              if(name.eq.'dt ') then
                        read*,dt
                        goto 1132
              endif
              if(name.eq.'tre') then
                        read*,trepo
                        goto 1132
              endif
              if(name.eq.'tto') then
                        read*,ttot
                        if(trepo.lt.0) trepo= -ttot
                        goto 1132
              endif
              if(name.eq.'cor') then
                        read*,coriof
                        goto 1132
              endif
              if(name.eq.'coa') then
                        read*,corioa
                        goto 1132
              endif
              if(name.eq.'cob') then
                        read*,coriob
                        goto 1132
              endif
```

```
              if(name.eq.'g   ') then
                      read*,g
                      goto 1132
              endif
              if(name.eq.'u0 ') then
                      read*,u0
                      goto 1132
              endif
              if(name.eq.'v0 ') then
                      read*,v0
                      goto 1132
              endif
              if(name.eq.'h0 ') then
                      read*,h0
                      goto 1132
              endif
              if(name.eq.'top') then
                      read*,top
                      goto 1132
              endif
              if(name.eq.'wid') then
                      read*,width
                      goto 1132
              endif
      stop
      endif
9999      continue
      trepo=abs(trepo)
      print*,'dt,treport,ttotal='
     ,,dt,trepo,ttot
      print*,'coriolis f,corioa,coriob,g='
     ,,coriof,corioa,coriob,g
      print*,'u0,v0,top,width,h0='
     ,,u0,v0,top,width,h0
      do 2 i=1,lparm0
2      print*,parm0(i)
      return
      end
```

**3. Node program**

```
      PROGRAM NODE
c
      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
      parameter(nodes=-1,idhost=2,nodepid=3)

c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx - 1)
      parameter(i9dim=4+(1+i9global)/nprocs)
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c


c
c The array UVH  contains the data u,v,h, in a format that allows
c fast message passing.
c At a fixed i, just send a buffer beginning at UVH(0,1,i)
c with length 2(columns)*3(variables)*(j9+1) words
c
      parameter(lenUVH =6*(j9+1)*4 )
c                       4 bytes per real
      parameter(inddt0=9140,inddt9=9149)
      parameter(ku=1,kv=2,kh=3)
      common/allcom/ UVH(0:j9,3,0:i9dim)
     _,zeta(0:i9dim,0:j9),hq(0:i9dim,0:j9),q(0:i9dim,0:j9)
     _,f(0:i9dim,0:j9),alfa(0:i9dim,0:j9),beta(0:i9dim,0:j9)
     _,gama(0:i9dim,0:j9),delta(0:i9dim,0:j9)
     _,eps(0:i9dim,0:j9),fi(0:i9dim,0:j9)
     _,cay(0:i9dim,0:j9),ustar(0:i9dim,0:j9),vstar(0:i9dim,0:j9)
     _,dudt(0:i9dim,0:j9),dvdt(0:i9dim,0:j9),dhdt(0:i9dim,0:j9)
     _,hs(0:i9dim,0:j9),hu(0:i9dim,0:j9),hv(0:i9dim,0:j9)

      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax

      common/dtcom/dt,trepo,ttot

      common/physcom/coriof,corioa,coriob,g

      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      common/seqcom/lmnpr
```

```
c
c during debug only
c
      dimension uold(0:i9dim,0:j9)
     _,vold(0:i9dim,0:j9),hold(0:i9dim,0:j9)

c
c during debug only
c
      if(mynode().ge.nprocs) stop
      iam=mynode()
      lmnpr=1000  +100*iam

      m0dul0=i9global+1
      call init
      nreport=trepo/dt
      nstep9=ttot/dt
      dthalf=dt/2
      twodt=2*dt

      nstep=0
      call report(nstep,nreport,nstep9)


c=================================================
c sending data to the neighbors THE VERY FIRST TIME
c the data get to different buffers, so the CALLs are synchronous
c
      msgbefo=isend(inddt9,UVH(0,1,imin),lenUVH,ibefore,nodepid )
      msgaftr=isend(inddt0,UVH(0,1,imax-1),lenUVH,iafter,nodepid )
c=================================================

10      continue
c
c Heun step
c
      call defddt

c
c until now, UVH has not changed. before it changes,
c ensure the sending worked
c
      call msgwait(msgbefo)
      call msgwait(msgaftr)

      do 11 i=0,i9
      do 11 j=0,j9
      uold(i,j)=UVH(j,ku,i)
      UVH(j,ku,i) =uold(i,j)+dudt(i,j)*dthalf
      vold(i,j)=UVH(j,kv,i)
      UVH(j,kv,i) =vold(i,j)+dvdt(i,j)*dthalf
      hold(i,j)=UVH(j,kh,i)
```

```
      UVH(j,kh,i) =hold(i,j)+dhdt(i,j)*dthalf
11      continue
c================================================
c
c the step ends with sending data to the neighbors
c the data get to different buffers, so the CALLs are synchronous
c
      msgbefo=isend(inddt9,UVH(0,1,imin),lenUVH,ibefore,nodepid )
      msgaftr=isend(inddt0,UVH(0,1,imax-1),lenUVH,iafter,nodepid )
      call defddt
c
c until now, UVH has not changed. before it changes,
c ensure the sending worked
c
      call msgwait(msgbefo)
      call msgwait(msgaftr)
c================================================
      do 12 i=0,i9
      do 12 j=0,j9
      UVH(j,ku,i) =uold(i,j)+dudt(i,j)*dt
      UVH(j,kv,i) =vold(i,j)+dvdt(i,j)*dt
      UVH(j,kh,i) =hold(i,j)+dhdt(i,j)*dt
12      continue
c================================================
c
c the step ends with sending data to the neighbors
c the data get to different buffers, so the CALLs are synchronous
c
      msgbefo=isend(inddt9,UVH(0,1,imin),lenUVH,ibefore,nodepid )
      msgaftr=isend(inddt0,UVH(0,1,imax-1),lenUVH,iafter,nodepid )
      nstep=nstep+1
      call report(nstep,nreport,nstep9)
125     continue
c
c following leapfrog steps
c
      call defddt
c
c until now, UVH has not changed. before it changes,
c ensure the sending worked
c
      call msgwait(msgbefo)
      call msgwait(msgaftr)
c================================================
      do 13 i=0,i9
      do 13 j=0,j9
      temp=UVH(j,ku,i)
      UVH(j,ku,i) =uold(i,j)+dudt(i,j)*twodt
      uold(i,j)=temp
      temp=UVH(j,kv,i)
      UVH(j,kv,i) =vold(i,j)+dvdt(i,j)*twodt
      vold(i,j)=temp
```

```
      temp=UVH(j,kh,i)
      UVH(j,kh,i) =hold(i,j)+dhdt(i,j)*twodt
      hold(i,j)=temp
13      continue
c================================================
c
c the step ends with sending data to the neighbors
c the data get to different buffers, so the CALLs are synchronous
c
        msgbefo=isend(inddt9,UVH(0,1,imin),lenUVH,ibefore,nodepid )
        msgaftr=isend(inddt0,UVH(0,1,imax-1),lenUVH,iafter,nodepid )
c================================================
      nstep=nstep+1
      call report(nstep,nreport,nstep9)
      if( mod(nstep+1,300).eq.1) then
      goto 10
      else
      goto 125
      endif
      end
      subroutine UVHpr(UVH,t,kUVH, leadim,i9,j9 )
      parameter(ku=1,kv=2,kh=3)
      character*(*) t
      common/seqcom/lmnpr
      dimension UVH(0:j9,3,0:leadim)
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i90,i8,imin,imax
c
c comment out the return during debug
c
      return
      print2000,10000*lmnpr,t,leadim,i9,j9,imin,imax,iam
2000      format(i10,2x,a20,' leadim,i9,j9=',3i4,' imin,imax=',3i4)
      do 1 i=0,i9
      ii=mod(i+i0global ,m0dul0)
      if(i.ge.imin.and.i.le.imax) then
      iamy=iam
      else
      iamy=iam+10
      endif
      do 1 j=0,j9
      print1000,ii+100*(j+100*lmnpr),t,j,ii
     _,UVH(j,kUVH, i),iamy
1      continue
1000      format(i10,2x,a10,2x,2i4,f20.7,i3)
      lmnpr=lmnpr+1
      return
      end
```

```
      subroutine bokepn(b00kp)
c
c the node sees its data as an array (0:i9dim,0:j9)
c
c the column (,i) in the node data is the same as (,i+i0global) in
c the full matrix, which is (0:i9dimglobal,0:j9)
c
c most of the work was done by the host, and
c data is just rearranged here
c
      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
c connection parameters
      parameter( myslv=1,ibev=2,iav=3,i0gv=4)
      parameter( iminv=5,imaxv=6,i9v=7)

      dimension b00kp(10,0:node9)
      common/seqcom/lmnpr
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax

      iam=mynode()
      myslice=b00kp(myslv,iam)
      ibefore=b00kp(ibev,iam)
      iafter=b00kp(iav,iam)
      i0global=b00kp(i0gv,iam)
      i9=b00kp(i9v,iam)
      imin=b00kp(iminv,iam)
      imax=b00kp(imaxv,iam)
      i8=i9-1
c
c print during debug only
c
cdebug      key=lmnpr*10000+100*iam
cdebug      print*,key,' ... I am ',iam
cdebug      key=key+1
cdebug      print*,key,' slice=', myslice
cdebug      key=key+1
cdebug      print*,key,' nodes bef,aft=',ibefore,iafter
cdebug      key=key+1
cdebug      print*,key,' data dim (0:' ,i9,',)'
cdebug      key=key+1
cdebug      print*,key,'      ',imin, ' to ',imax,' meaningful'
cdebug      key=key+1
cdebug      print*,key,' my column 0=global ',i0global
cdebug      lmnpr=lmnpr+1
      return
      end
```

```
      function coriofun(i,j,meshx,meshy)
      parameter(lparm0=10)
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax
      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      x=meshx*mod(i0global+i,m0dul0)
      y=meshy*(j)
c                          x,y for f --like zeta-- from C-mesh
      coriofun=coriof
c
c for the simplest case, coriolis is constant, but in general
c it might be computed using x,y and the parm0 data
c or corioa, coriob -- tangent plane, if needed
c
      return
      end
      subroutine defddt
c
c the results always in dudt, dvdt, dhdt in aLlcom
c

      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx-1)
      parameter(i9dim=4+(1+i9global)/nprocs)
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c



c
c The array UVH contains the data u,v,h, in a format that allows
c fast message passing.
c At a fixed j, just send a buffer beginning at UVH(0,1,j)
c with length 6*(i9+1) words = 2 rows of 3 vars each
      parameter(lenUVH =6*(j9+1)*4 )
c                         4 bytes per real
      parameter(inddt0=9140,inddt9=9149)
c
      parameter(ku=1,kv=2,kh=3)
```

```
       common/allcom/ UVH(0:j9,3,0:i9dim)
     _,zeta(0:i9dim,0:j9),hq(0:i9dim,0:j9),q(0:i9dim,0:j9)
     _,f(0:i9dim,0:j9),alfa(0:i9dim,0:j9),beta(0:i9dim,0:j9)
     _,gama(0:i9dim,0:j9),delta(0:i9dim,0:j9)
     _,eps(0:i9dim,0:j9),fi(0:i9dim,0:j9)
     _,cay(0:i9dim,0:j9),ustar(0:i9dim,0:j9),vstar(0:i9dim,0:j9)
     _,dudt(0:i9dim,0:j9),dvdt(0:i9dim,0:j9),dhdt(0:i9dim,0:j9)
     _,hs(0:i9dim,0:j9),hu(0:i9dim,0:j9),hv(0:i9dim,0:j9)

       common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax

       common/dtcom/dt,trepo,ttot

       common/physcom/coriof,corioa,coriob,g

       common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

       i7=i8-1
       dx=meshx
       dy=meshy
       j8=j9-1
       j7=j9-2
c================================================
c
c any such step begins with receiving data from the neighbors
c the data get to different buffers, so the CALLs are synchronous
c
         in0=irecv(inddt0,UVH(0,1,0),lenUVH )
         in9=irecv(inddt9,UVH(0,1,i8),lenUVH )
        call msgwait(in0)
        call msgwait(in9)
             CALL UVHpr(UVH,'u after irecv',ku,i9dim,i9,j9)
             CALL UVHpr(UVH,'v after irecv',kv,i9dim,i9,j9)
             CALL UVHpr(UVH,'h after irecv',kh,i9dim,i9,j9)

c
c The big mess is that u,v,h are not defined everywhere!
c            u defined for 0<=i<=i9, 0<=j < j9
c            v defined for 0<=i < i9, 0<=j<=j9
c            h defined for 0<=i < i9, 0<=j < j9
c
c But because of periodicity in i , which is maintained by the
c send+receive :
c            v defined for 0<=i<=i9, 0<=j<=j9
c            h defined for 0<=i<=i9, 0<=j < j9
c
c So, finally, only the following are MISSING:
c            u(,j9) , h(,j9)
```

```
c
c NOW WE DEFINE ALL VARIABLES WHEREVER POSSIBLE
c    THEN,
c       SEE IF d/dt IS DEFINED WHERE IT SHOULD:
c            imin<=i<=imax,  jmin<=j<=jmax
c
c       imin,imax from bookkeeping,
c       jmin,jmax depend on the variable u,v,h
c
c IT TURNS OUT THAT EVERYTHING IS SAFE PROVIDED:  2<=imin , imax < i8
c
c       N.B. imax less than i8 !
c


c
c 3.12 in the paper follows
c
c
      do 3120 i=1,i9
      do 3121 j=1,j8
c
c both i,j safe
c
       zeta(i,j)=(UVH(j-1,ku,i) - UVH(j,ku,i))/dy
     _+(UVH(j,kv,i) -UVH(j,kv,i-1))/dx
3121       continue
c
c THE PAPER HAS THE COMPUTATIONAL BD.CD. ZETA=0
c
      zeta(i,0)=0
      zeta(i,j9)=0
3120       continue
312       continue
cdebug             CALL matpr('zeta',zeta,i9dim,i9,j9)


c
c 3.15 in the paper follows
c
c
      do 3150 j=1,j8
      do 3151 i=1,i9
c
c both i,j safe, periodic bd.cd in i automatic
c
      hq(i,j)=(UVH(j,kh,i)+UVH(j,kh,i-1)+UVH(j-1,kh,i-1)
     ++UVH(j-1,kh,i))/4
3151       continue
3150       continue
```

```
c
c this still leaves hq undefined at j=0
c
c do some extrapolation, maybe zeta=0 will take care of
c instability
c
      do 3156 i=1,i9
      hq(i,0)=3*hq(i,1)-3*hq(i,2)+hq(i,3)
3156       continue
c
c this still leaves hq undefined at j=j9
c
c do some extrapolation, maybe zeta=0 will take care of
c instability
c
      do 3157 i=1,i9
      hq(i,j9)=3*hq(i,j9-1)-3*hq(i,j9-2)+hq(i,j9-3)
3157       continue
315      continue
cdebug             CALL matpr('hq',hq,i9dim,i9,j9)



c
c 3.11 in the paper follows
c
      do 311 i=1,i9
      do 311 j=0,j9
      q(i,j)=(f(i,j)+ zeta(i,j)) / hq(i,j)
311      continue
cdebug             CALL matpr('q',q,i9dim,i9,j9)
c
c 3.34 in the paper follows
c
      do 334 j=0,j8

c the same j loop all over

      do 3340 i=1,i8
      eps(i,j)=(q(i+1,j+1)+ q(i,j+1)-q(i,j)-q(i+1,j))/24
      fi(i,j)=(-q(i+1,j+1)+ q(i,j+1)+q(i,j)- q(i+1,j))/24
c
c these have indices i+1/2, j+1/2 only, like h
c
3340      continue

      do 3341 i=2,i8
      alfa(i,j)=(2*q(i+1,j+1)+q(i,j+1)+ 2*q(i,j)+ q(i+1,j))/24
      beta(i,j)=(q(i,j+1)+ 2*q(i-1,j+1)+ q(i-1,j)+ 2*q(i,j))/24
      gama(i,j)=(2*q(i,j+1)+ q(i-1,j+1)+ 2*q(i-1,j)+ q(i,j))/24
      delta(i,j)=(q(i+1,j+1)+2*q(i,j+1)+ q(i,j)+2*q(i+1,j))/24
3341      continue
```

```
c closing the j loop

334       continue

cdebug       CALL matpr('eps',eps,i9dim,i9,j9)
cdebug       CALL matpr('fi',fi,i9dim,i9,j9)


cdebug       CALL matpr('alfa',alfa,i9dim,i9,j9)
cdebug       CALL matpr('beta',beta,i9dim,i9,j9)
cdebug       CALL matpr('gama',gama,i9dim,i9,j9)
cdebug       CALL matpr('delta',delta,i9dim,i9,j9)

c
c 3.36-3.37 in the paper follow
c
      do 336 j=0,j8
      do 336 i=1,i9
       hu(i,j)=(UVH(j,kh,i-1) +UVH(j,kh,i))/2
336     continue
      do 337 i=0,i9
      do 337 j=1,j9
       hv(i,j)=(UVH(j-1,kh,i) + UVH(j,kh,i))/2
c
c at j=0 and j=j9, hv may be anything, because v=0
c
337       continue
cdebug       CALL matpr('hu',hu,i9dim,i9,j9)
cdebug       CALL matpr('hv',hv,i9dim,i9,j9)



c
c 3.41 in the paper follows
c
      do 341 i=0,i8
      do 341 j=0,j8
       cay(i,j)=(UVH(j,ku,i) **2 +UVH(j,ku,i+1) **2
     _+UVH(j,kv,i) **2 +UVH(j+1,kv,i) **2)/4
341       continue
cdebug       CALL matpr('cay',cay,i9dim,i9,j9)



c
c 3.3-3.4 in the paper follow
c
      do 33 j=0,j8
      do 33 i=1,i9
       ustar(i,j)= hu(i,j)* UVH(j,ku,i)
33      continue
```

```
      do 34 i=0,i9
      vstar(i,0)=0
c
c bd.cd       v=0
c
      do 34 j=1,j9
       vstar(i,j)= hv(i,j)* UVH(j,kv,i)
34      continue
cdebug        CALL matpr('ustar',ustar,i9dim,i9,j9)
cdebug        CALL matpr('vstar',vstar,i9dim,i9,j9)

c
c 3.1-3.2 in the paper follow
c
      do 32 i=1,i8
      do 32 j=0,j8
c
c this is where dhdt values are defined. they are needed for:
c      imin <= i <= imax
c      0    <= j <= j8
c            NOT needed for j=j9
c
      dhdt(i,j)=(ustar(i,j)-ustar(i+1,j))/dx
     _+(vstar(i,j)-vstar(i,j+1))/dy
31      continue
32      continue

cdebug        CALL matpr('dhdt',dhdt,i9dim,i9,j9)

c
c 3.5 in the paper follows to compute du/dt
c
c
      do 35 j=0,j8
      do 35 i=2,i8
c
c this is where dudt is defined . it is needed for:
c      imin <= i <= imax
c      0    <= j <= j8
c            NOT needed for j=j9
c
c 2.5 in the paper follows, defining capital phi in terms of h,hs
c
       caphi1=g*(UVH(j,kh,i-1) +hs(i-1,j))
       caphi2=g*(UVH(j,kh,i) +hs(i,j))

      dudt(i,j)= alfa(i,j)* vstar(i,j+1)+beta(i,j)* vstar(i-1,j+1)
     _+gama(i,j)* vstar(i-1,j)
     _+delta(i,j)* vstar(i,j)-eps(i,j)*ustar(i+1,j)
     _+eps(i-1,j)* ustar(i-1,j)
     _+(cay(i-1,j)+ caphi1 -cay(i,j)- caphi2)/dx
35      continue
```

```
cdebug       CALL matpr('dudt',dudt,i9dim,i9,j9)

c
c 3.6 in the paper follows to compute dv/dt
c
      do 36 i=2,i8-1
      do 36 j=1,j8
c
c this is where dvdt is defined
c it is needed for
c      imin <= i <= imax
c      1    <= j <= j8
c
c dvdt NOT needed at j=0 or j=j9, because  there v=0 always
c
c 2.5 in the paper follows, defining capital phi in terms of h,hs
c
       caphi3=g*(UVH(j-1,kh,i) +hs(i,j-1))
       caphi4=g*(UVH(j,kh,i) +hs(i,j))

      dvdt(i,j)=  -gama(i+1,j)* ustar(i+1,j)- delta(i,j)* ustar(i,j)
     _- alfa(i,j-1)*ustar(i,j-1)- beta(i+1,j-1)* ustar(i+1,j-1)
     _+fi(i,j-1)*vstar(i,j-1)+fi(i,j)* vstar(i,j+1)
     _+(caphi3+cay(i,j-1)-caphi4 - cay(i,j))/dy
36      continue
cdebug       CALL matpr('dvdt',dvdt,i9dim,i9,j9)
      return
      end
      function hsfun(i,j,meshx,meshy)
      parameter(lparm0=10)
      parameter(maxx=6000,maxy=2000)
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax
      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      x=meshx*(mod(i0global+i,m0dul0)+0.5)
      y=meshy*(j+0.5)
c                          x,y for h from C-mesh
      xx=abs(x-0.5*maxx)
      if(xx.lt.width) then
      hsfun=top*(1-xx/width)
      else
      hsfun=0
      endif
c
c this is the triangular ridge. more generally,
c hsfun might be computed using x,y and the parm0 data
c
      return
      end
```

```
      subroutine init

      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
      parameter(lbuf=10*nprocs+30+lparm0,leninit=lbuf*4)
c                               4 bytes per float
      parameter(inityp=914)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx - 1)
      parameter(i9dim=4+(1+i9global)/nprocs)
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c


      parameter(ku=1,kv=2,kh=3)
      common/allcom/ UVH(0:j9,3,0:i9dim)
     _,zeta(0:i9dim,0:j9),hq(0:i9dim,0:j9),q(0:i9dim,0:j9)
     _,f(0:i9dim,0:j9),alfa(0:i9dim,0:j9),beta(0:i9dim,0:j9)
     _,gama(0:i9dim,0:j9),delta(0:i9dim,0:j9)
     _,eps(0:i9dim,0:j9),fi(0:i9dim,0:j9)
     _,cay(0:i9dim,0:j9),ustar(0:i9dim,0:j9),vstar(0:i9dim,0:j9)
     _,dudt(0:i9dim,0:j9),dvdt(0:i9dim,0:j9),dhdt(0:i9dim,0:j9)
     _,hs(0:i9dim,0:j9),hu(0:i9dim,0:j9),hv(0:i9dim,0:j9)

      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax

      common/dtcom/dt,trepo,ttot

      common/physcom/coriof,corioa,coriob,g

      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      dimension buf(lbuf)

      call crecv(inityp,buf,leninit)
      call bokepn (buf)
      l=10*nprocs+1
      dt=buf(l)
      l=l+1
      trepo=buf(l)
      l=l+1
      ttot=buf(l)
      l=l+1
      coriof=buf(l)
      l=l+1
      corioa=buf(l)
```

```
      l=l+1
      coriob=buf(l)
      l=l+1
      g=buf(l)
      l=l+1
      u0=buf(l)
      l=l+1
      v0=buf(l)
      l=l+1
      top=buf(l)
      l=l+1
      width=buf(l)
      l=l+1
      h0=buf(l)
      l=l+1
      do 976 j=1,lparm0
      parm0(j)=buf(l)
      l=l+1
976      continue
c
c the common allcom is filled with trash, to check
c glitches in index manipulation
c
c but some variables get meaningful values: f,vstar,...
c
      do 1 i=0,i9
      do 1 j=0,j9
      UVH(j,ku,i) =1.e6+j+100*i
      zeta(i,j)=3.e6+j+100*i
      UVH(j,kh,i) =4.e6+j+100*i
      hq(i,j)=5.e6+j+100*i
      q(i,j)=6.e6+j+100*i
      f(i,j)=coriofun(i,j,meshx,meshy)
      alfa(i,j)=8.e6+j+100*i
      beta(i,j)=9.e6+j+100*i
      gama(i,j)=-1.e6-j-100*i
      delta(i,j)=-2.e6-j-100*i
      eps(i,j)=-3.e6-j-100*i
      fi(i,j)=-4.e6-j-100*i
      hu(i,j)=-5.e6-j-100*i
      hv(i,j)=-6.e6-j-100*i
      cay(i,j)=-7.e6-j-100*i
      ustar(i,j)=-8.e6-j-100*i
      UVH(j,kv,i) =0
      vstar(i,j)=0
c
c implementation of WALL bd.cd
c
      dudt(i,j)=0
      dvdt(i,j)=0
      dhdt(i,j)=0
1      continue
```

```
c
c initial data, defined on PART OF the arrays
c

      j8=j9-1
      do 170 i=0,i9
      do 170 j=0,j8
      UVH(j,ku,i) =u0fun(i,j,meshx,meshy )
170      continue
      do 171 i=0,i9
      do 171 j=1,j8
c
c 1<=j<=j8   <=>  implementation of WALL bd.cd
c
      UVH(j,kv,i) =v0fun(i,j,meshx,meshy )
171      continue
      do 172 i=0,i9
      do 172 j=0,j9
      hs(i,j)=hsfun(i,j,meshx,meshy)
c
c hs is defined everywhere, it is the geography
c
172      continue
      do 271 i=0,i9
      do 271 j=0,j8
      UVH(j,kh,i) =h0-hs(i,j )
271      continue

       CALL UVHpr(UVH,'u init',ku,i9dim,i9,j9)
       CALL UVHpr(UVH,'v init',kv,i9dim,i9,j9)
       CALL UVHpr(UVH,'h init',kh,i9dim,i9,j9)


      return
      end
      subroutine matpr(t,a,leadim, i9 ,j9)
      common/seqcom/lmnpr
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i90,i8,imin,imax
      character*(*) t
      dimension a(0:leadim,0:j9)
c
c comment out the return during debug
c
      return
      print2000,10000*lmnpr,t,leadim,i9,j9,imin,imax,iam
2000      format(i10,2x,a10,' leadim,i9,j9=',3i4,' imin,imax=',3i4)
      do 1 i=0,i9
      ii=mod(i+i0global ,m0dul0)
      if(i.ge.imin.and.i.le.imax) then
      iamy=iam
      else
```

```
      iamy=iam+10
      endif
      do 1 j=0,j9
      print1000,ii+100*(j+100*lmnpr),t,j,ii,a(i,j),iamy
1      continue
1000      format(i10,2x,a10,2x,2i4,f20.7,i3)
      lmnpr=lmnpr+1
      return
      end
      subroutine report(n,nrep,nmax)
c
c n is the number of steps
c

      parameter(lparm0=10)

c cube parameters
      parameter(nprocs=8,node9=nprocs-1)
      parameter(nodes=-1,idhost=2,nodepid=3)
c domain constants
      parameter(maxx=6000,maxy=2000,meshy=50,meshx=50)
      parameter(j9=maxy/meshy,i9global=maxx/meshx - 1)
      parameter(i9dim=4+(1+i9global)/nprocs)
c
c notice that the domain size and mesh define the
c dimensions of all the arrays
c

c
c The array UVH contains the data u,v,h, in a format that allows
c fast message passing.
c
      parameter(len=12*(i9dim+1)*(j9+1))
      parameter(ku=1,kv=2,kh=3)
      common/allcom/ UVH(0:j9,3,0:i9dim)
     _,zeta(0:i9dim,0:j9),hq(0:i9dim,0:j9),q(0:i9dim,0:j9)
     _,f(0:i9dim,0:j9),alfa(0:i9dim,0:j9),beta(0:i9dim,0:j9)
     _,gama(0:i9dim,0:j9),delta(0:i9dim,0:j9)
     _,eps(0:i9dim,0:j9),fi(0:i9dim,0:j9)
     _,cay(0:i9dim,0:j9),ustar(0:i9dim,0:j9),vstar(0:i9dim,0:j9)
     _,dudt(0:i9dim,0:j9),dvdt(0:i9dim,0:j9),dhdt(0:i9dim,0:j9)
     _,hs(0:i9dim,0:j9),hu(0:i9dim,0:j9),hv(0:i9dim,0:j9)

      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax

      common/dtcom/dt,trepo,ttot

      common/physcom/coriof,corioa,coriob,g

      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)
```

```
      if(mod(n,nrep).ne.0) return
      itype=n*100+iam
452      continue
      myh=myhost()
      call csend(itype,UVH,len,myh,idhost )
      if(n.ge.nmax) stop 5144
      return
      end
      function u0fun(i,j,meshx,meshy)
      parameter(lparm0=10)
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax
      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      x=meshx*mod(i0global+i,m0dul0)
      y=meshy*(j+0.5)
c                          x,y for u from C-mesh
      u0fun=u0
c
c for the simplest case, u is constant, but in general
c it might be computed using x,y and the parm0 data
c
      return
      end
      function v0fun(i,j,meshx,meshy)
      parameter(lparm0=10)
      common/bkkpcom/m0dul0,iam,myslice,ibefore,iafter
     _,i0global,i9,i8,imin,imax
      common/dtcom/dt,trepo,ttot
      common/physcom/coriof,corioa,coriob,g
      common/UVH0com/u0,v0,top,width,h0,parm0(lparm0)

      x=meshx*(mod(i0global+i,m0dul0)+0.5)
      y=meshy*(j)
c                          x,y for v from C-mesh
      v0fun=v0
c
c for the simplest case, v is constant, but in general
c it might be computed using x,y and the parm0 data
c
      return
      end
```

**4. Makefile**

```
# This file is used to compile and link the host.f, node.f
#
# The command "make all" causes compilation and linking.

all :   host node


host.o: host.f

node.o: node.f

host:   host.o
        f77 -o host host.o -host


node:   node.o
        f77 -o node node.o -node
```

## 5. Input File

```
dt
60
ttotal
79800
top
2
width
1000
u0
20.e-3
end
```

**Acknowledgements**

This research was conducted for the Office of Naval Research and was funded by the Naval Postgraduate School.

**References**

1. A. Arakawa, and V. R. Lamb, A potential Enstrophy and Energy Conserving Scheme for the Shallow Water Equations, Mon. Wea. Rev., Vol. 109 (1981),18-36.

2. B. Neta, and L. Lustman, Parallel Conservative Scheme for Solving the Shallow Water Equations, submitted for publication.

DISTRIBUTION LIST

| | No. of Copies |
|---|---|
| Director<br>Defense Tech. Inf. Center<br>Cameron Station<br>Alexandria, VA 22314 | 2 |
| Director of Research Admin.<br>Code 012<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| Library<br>Code 0142<br>Naval Postgraduate School<br>Monterey, CA 93943 | 2 |
| Dept. of Mathematics<br>Code MA<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |
| Center for Naval Analysis<br>4401 Ford Avenue<br>Alexandria, VA 22302-0268 | 1 |
| Professor Beny Neta<br>Code MA/Nd<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, CA 93943 | 15 |
| Professor Naotaka Okamoto<br>Okayama University of Science<br>Dept. of Applied Science<br>Ridai-cho 1-1, Okayama 700<br>Japan | 1 |
| Professor William Gragg<br>Code MA/Gr<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, CA 93943 | 5 |
| Professor Levi Lustman<br>NOARL<br>Monterey, CA 93943 | 15 |
| Dr. C.P. Katti<br>J. Nehru University<br>School of Computer<br>and Systems Sciences<br>New Delhi 110067<br>India | 1 |